PIC 20A

Yuanyuan Fu, Zhuo Kang, Ruoyu Liu & Feng Lu

PacMan Remaking in Java

Introduction

- Game Features
  - Maze selections
  - Power pills to speed up
  - Tunnels to move across the screen
- Winning Condition
  - Eat up all the pills
  - Stay alive in limited time
  - Avoid monsters

Game Engine

- Provides framework for creating games.
- The one we are using:
  - Game.java
    - **abstract** class which implements KeyListener, Mouse Listener and etc.
    - abstract method: **update() & draw(Graphics 2D g)**
  - Game Application.java
    - only has one method: start(Game game)
    - works like a constructor. Only need to put this in main.
      - **GameApplication.start(new PacMan());**
  - Game Canvas.java (for GUIs)
    - extends JComponent implements ComponentListener
  - GameLoop.java
    - extends Thread
    - inside run() it has **game.update() & canvas.repaint()**

Our Project

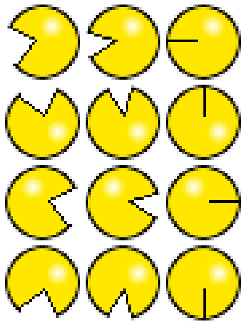
- PacMan.java
- Maze.java
- MazeConstructor.java (Helper class)

PacMan.java

- extends Game
- keyboard control

```
@Override
public void keyPressed(KeyEvent e){
    int key = e.getKeyCode();
    if(37<= key && key <= 40){
        reqDir = key;
    }
}
```

- animation

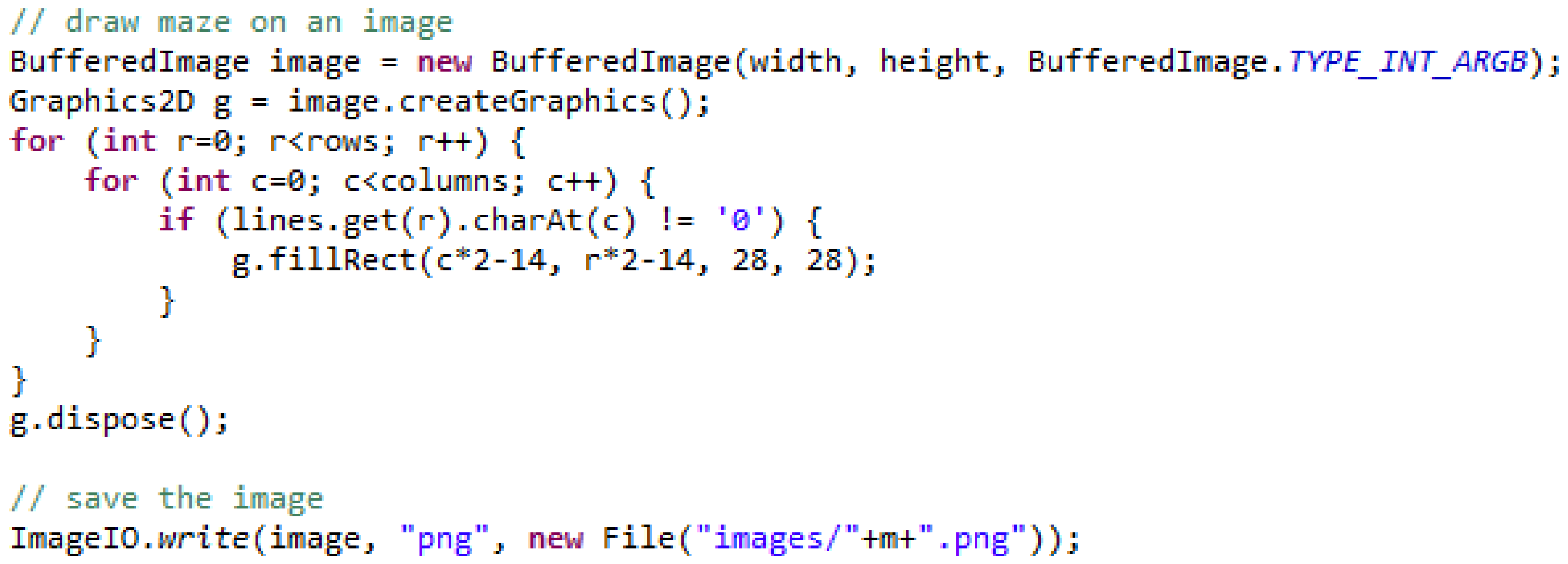


```
@Override
public void update() {
    frame++;
    if(frame > 5){
        frame = 0;
    }
```

```
@Override
public void draw(Graphics2D g) {
    g.drawImage(packman.getSubimage((frame/2)*30, (ableDir-37)*30, 28, 28),
            x*STEP-14, y*STEP-14, null);
```

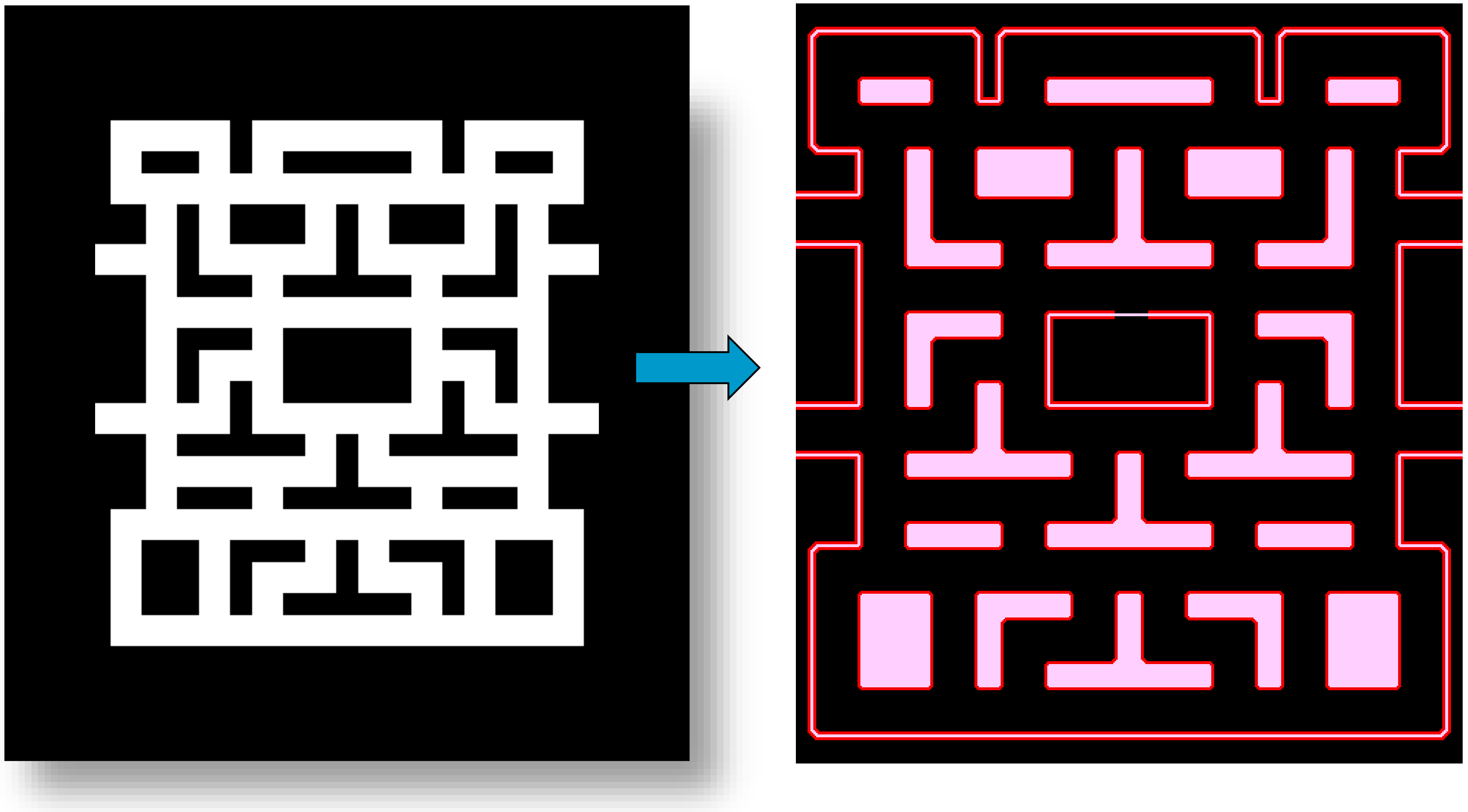
Maze Set-up

- a text file consisting of 0’s(walls), 1’s(roads), 2’s(pills) & 3’s (power pills).
- But we also have to put a corresponding layer of maze image on the canvas for visualization.



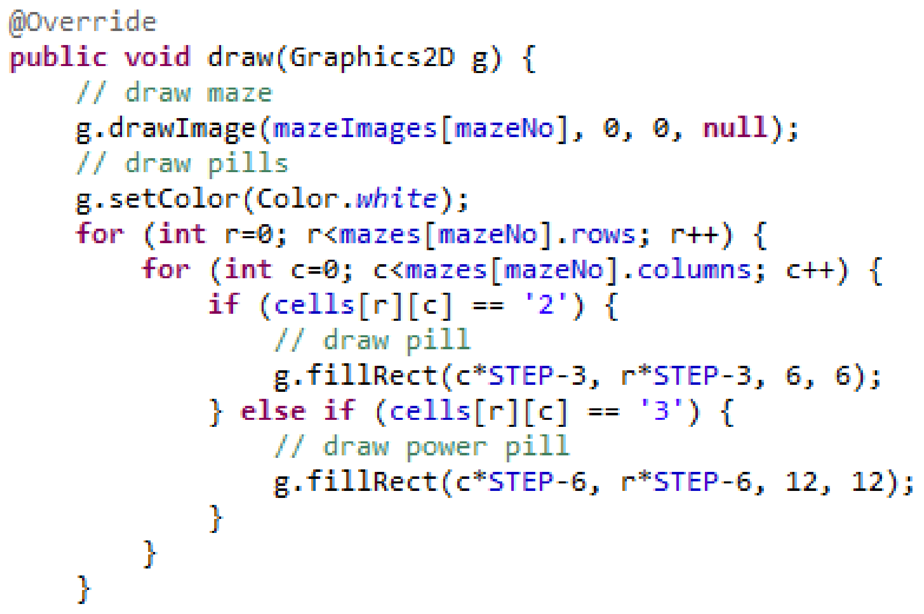
```
// draw maze on an image
BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_ARGB);
Graphics2D g = image.createGraphics();
for (int r=0; r<rows; r++) {
    for (int c=0; c<columns; c++) {
        if (lines.get(r).charAt(c) != '0') {
            g.fillRect(c*2-14, r*2-14, 28, 28);
        }
    }
}
g.dispose();

// save the image
ImageIO.write(image, "png", new File("images/"+m+".png"));
```

Maze Set-up (con't)



Maze Set-up (con’t)

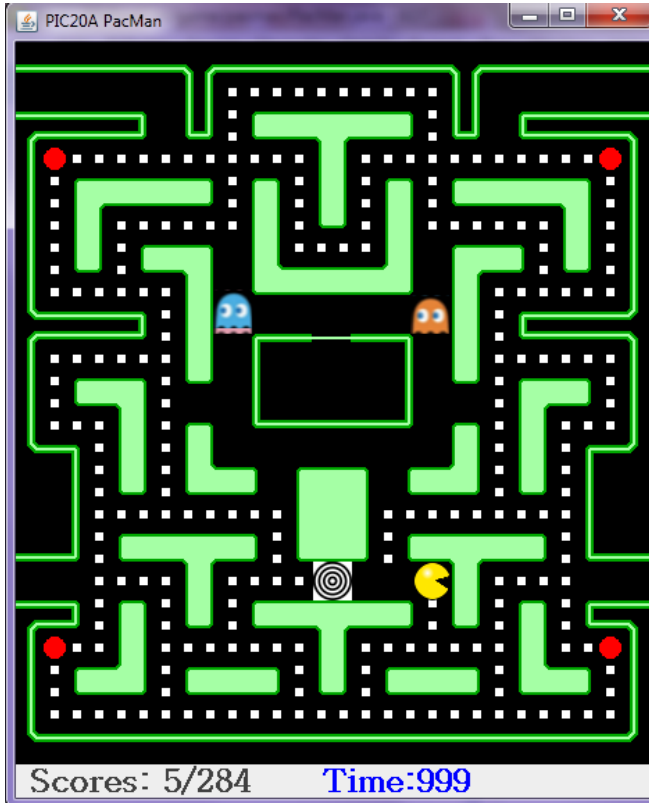
- Draw maze and pills



```
@Override
public void draw(Graphics2D g) {
    // draw maze
    g.drawImage(mazeImages[mazeNo], 0, 0, null);
    // draw pills
    g.setColor(Color.white);
    for (int r=0; r<mazes[mazeNo].rows; r++) {
        for (int c=0; c<mazes[mazeNo].columns; c++) {
            if (cells[r][c] == '2') {
                // draw pill
                g.fillRect(c*STEP-3, r*STEP-3, 6, 6);
            } else if (cells[r][c] == '3') {
                // draw power pill
                g.fillRect(c*STEP-6, r*STEP-6, 12, 12);
            }
        }
    }
```

Movement

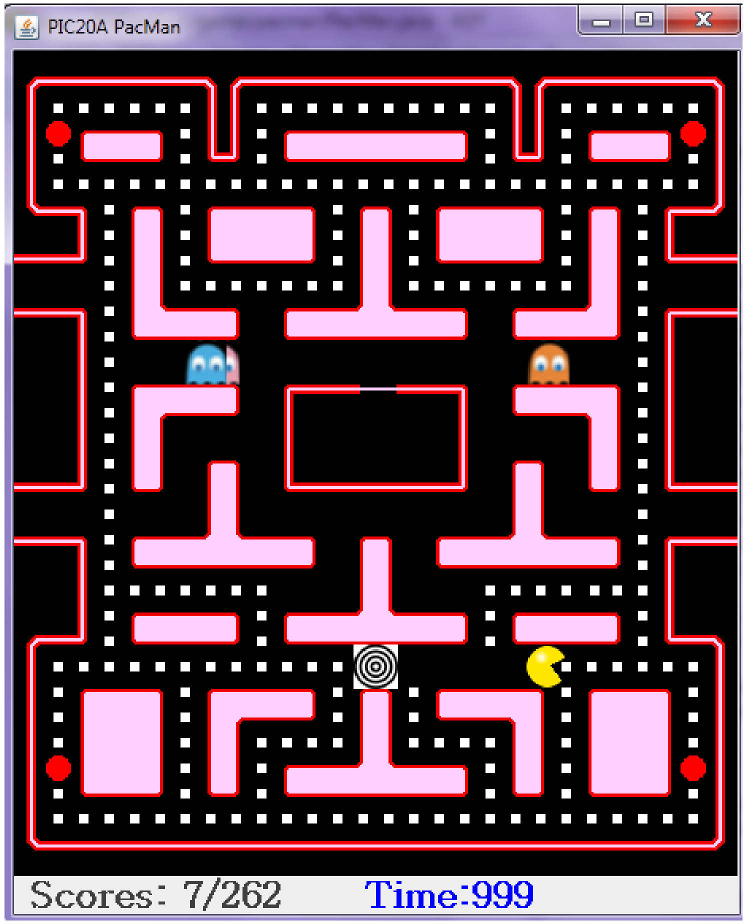
```
private int move(int reqDir) {
    switch (reqDir) {
    case KeyEvent.VK_LEFT: // 37
        if (x > 0 && mazes[mazeNo].charAt(y, x-1) != '0') {
            x -= 1;
            return SUCCESS;
        }
        if (x == 0 && copy[y][columns-1] == '1' ) {
            x = columns-1;
            return SUCCESS;
        }
        break;
```



```
// if the the request direction is possible, change the direction of pacman
// otherwise, stick with the current direction
if (move(reqDir) == SUCCESS) {
    ableDir = reqDir;
} else {
    move(ableDir);
}
```

Demo

```
timeCount++;
if (timeCount >= 1000/delay){
    time--;
    if (time <= timeLimit/2){
        M2reqDir = (int)Math.floor(Math.random()*4);
        M3reqDir = (int)Math.floor(Math.random()*4);
        M4reqDir = (int)Math.floor(Math.random()*4);
        M5reqDir = (int)Math.floor(Math.random()*4);
        M6reqDir = (int)Math.floor(Math.random()*4);
        M7reqDir = (int)Math.floor(Math.random()*4);
        M8reqDir = (int)Math.floor(Math.random()*4);
    }

    if (time <= 0){
        timeOut = true;
        over = true;
    }
    timeCount = 0;
}
```


PIC20A PacMan
Scores: 7/262
Time:999